Atomic Energy Control Board
P.O. Box 1046
Ottawa, Canada
K1P 5S9

Commission de contrôle de l'énergie atomique
C.P. 1046
Ottawa, Canada
K1P 5S9

INFO-0201

COMPUTER SOFTWARE QUALITY ASSURANCE

by

Kevin A. Ives
PRIOR Data Sciences Ltd.

A research report prepared for the
Atomic Energy Control Board
Ottawa, Canada

June 1986

Canada

Research report

COMPUTER SOFTWARE QUALITY ASSURANCE

A report prepared by Kevin A. Ives of PRIOR Data Sciences Ltd., under contract to the Atomic Energy Control Board.

ABSTRACT

The author defines some criteria for the evaluation of software quality assurance elements for applicability to the regulation of the nuclear industry. The author then analyses a number of software quality assurance standards. The major extracted SQA elements are then discussed, and finally specific software quality assurance recommendations are made for the nuclear industry.

RÉSUMÉ

L'auteur établit des critères pour l'évaluation des éléments de l'assurance-qualité des logiciels et de leur application possible à la réglementation de l'industrie nucléaire. Il analyse ensuite certaines normes d'assurance-qualité des logiciels, traite des principaux éléments qui en ressortent et formule enfin des recommandations précises au sujet de l'assurance-qualité des logiciels à l'intention de l'industrie nucléaire.

Disclaimer

The Atomic Energy Control Board is not responsible for the accuracy of the statements made or opinions expressed in this publication and neither the Board nor the author assume liability with respect to any damage or loss incurred as a result of the use made of the information contained in this publication.

June, 1986

TABLE OF CONTENTS

ABSTRACT i

A. INTRODUCTION 1

1. Background Information
2. Report Objective
3. Report Format

B. STANDARDS EVALUATION CRITERIA 4

C. THE STANDARDS AND THEIR ASSESSMENT 7

1. CSA Preliminary Standard Q396.1 - 1982 8
Software Quality Assurance Program, Part 1

2. BS 5887: 1980 Testing of Computer-based Systems . 10

3. BS 6238: 1982 Performance Monitoring of 13
Computer-based Systems

4. BS 5515: 1984 Documentation of 15
Computer-based Systems

5. BS 6488: 1984 Configuration Management of 18
Computer-based Systems

6. Department of National Defence System 20
Development Methodology (DND SDM)

7. CAN3-N286.1-84 Procurement Quality Assurance . . 22
for Nuclear Power Plants

8. CSA N286.2-1979 Preliminary Standard, Design . . 25
Quality Assurance for Nuclear Power Plants

9. FAA-STD-018 May 1977, Federal Aviation 27
Administration, Computer Software Quality
Program Requirements

10. DGMEM SOFTWARE STANDARDS, 11-1979, Produced . . . 30
by The Canadian Forces

11. ANSI/IEEE Std 730-1984. Software Quality Assurance Plans 34

12. ANSI/IEEE Std 828-1983. Software Configuration Management Plans . 37

13. ANSI/IEEE Std 829-1983. Software Test Documentation 40

14. ANSI/IEEE Std 830-1984. Guide to Software Requirements Specifications 43

15. A.P.E.O. Guidelines to Standards of Practice for the use of Computer Programs in Engineering . . 45

16. DOD-STD-2167, June 1985. Defense System Software Development 47

17. MIL-S-52779A, Aug 1979. Software Quality Assurance Program Requirements 50

18. DOD-STD-1679A(Navy), Oct 1983. Software Development 53

D. EXTRACTION OF MAJOR REQUIRED ELEMENTS 57

1. Management
2. Engineering
3. Testing
4. Operational and Support

E. STANDARDS AND PRACTICES RECOMMENDATION 64

1. General Recommendations
2. Specific SQA Recommendations

2.1 Management
2.2 Engineering
2.3 Testing
2.4 Operational and Support

REFERENCES 72

BIBLIOGRAPHY 73

APPENDICES A. ABBREVIATIONS A1

B. ADDITIONAL STANDARDS AND RELATED DOCUMENTS . . . B1

A. INTRODUCTION

1. Background Information

Computing systems have always been an integral part of the Canadian nuclear industry, both for computational and analysis purposes, and in the areas of reactivity and process control. In recent years, the drastic reduction of computer hardware costs and the greater demands for efficiency and flexibility of operation, have all conspired to increase the application of computer control, modelling and analysis within the nuclear industry. This is particularly true of such areas as safety system operation and analysis, plant and operator-training simulators, and radioactive waste management and disposal modelling.

The enormous growth in the size and complexity of computer programs in nuclear applications has resulted in an increased requirement for the Atomic Energy Control Board (AECB) to perform knowledgeable and efficient assessments of the quality, reliability and safety of the computer programs referenced within licensee submissions. Hence there is increasing attention being given within the AECB to the subject of software quality assurance, and a growing need for AECB staff to have a sound knowledge of this topic.

AECB staff are therefore currently developing criteria for judging the methods being proposed by licensees and applicants for the assurance of quality in the design, coding, documentation, verification, installation, validation and maintenance of computer software used in reactivity, process, and safety systems in nuclear power stations. They are also participating in nuclear industry standards-writing activities related to software quality assurance.

It was therefore felt that a comprehensive survey and analysis of available software quality assurance (SQA) standards and practices current within the computer industry at large, would be of great value in the assessment of the methods proposed by licensees and applicants.

2. Report Objective

The objective of this report is therefore to provide a documented survey and assessment of the various standards and practices for computer software quality assurance in current use in Canada and other countries, and to provide guidance on the various elements of such standards and practices considered appropriate in ensuring the required quality for the software of certain applications within the nuclear industry.

The particular applications or subdivisions of software usage that were considered when assessing SQA elements are:

a. real-time software in nuclear power stations
   i.e. - Reactivity and process control systems
   - safety shutdown systems
   - emergency core-cooling and containment control
   - data acquisition and display systems

b. computer-aided system design facilities (for nuclear power plant control software).

Note that this report is not intended to be a criticism of, or comment on, current nuclear industry practices. In fact, no recognition has been taken of current SQA practices within the industry, other than the particular standards examined within this report. Indeed, many of the recommended SQA elements may already be common practice within the industry.

3. Report Format

The following section B of this report presents the evaluation criteria by which the applicability of the various software quality assurance standards, practices and elements were judged. The evaluation criteria were developed with regard to the terms of reference of the Atomic Energy Control Board and the above-mentioned software areas.

Section C constitutes the bulk of the report and contains a comprehensive assessment of a number of diverse standards and practices currently used in the software development industry in Canada and other countries. These standards are assessed in the light of what is currently regarded as 'good practice' within the SQA community, as well as by the section B evaluation criteria.

Section D presents a discussion of major software quality assurance elements extracted from the section C standards assessment activity. It was felt that an attempt to discuss or explain the elements to any depth was beyond the scope of this report. Accordingly, this section should be considered as a checklist of items to be evaluated by the criteria of section B.

Finally section E presents the specific recommendations of the author with regard to software quality assurance practices considered appropriate and applicable to the nuclear power industry. These SQA practices arise of course, from using the evaluation criteria of section B to filter the standards and elements of sections C and D.

The recommendations of this report have been made with little regard to the current or proposed activities of AECB licensees, and hence are not intended to imply any dificiencies in such practices. Rather this report is intended to provide an experienced but independent look at the subject of software quality assurance as it applies to the real-time software in nuclear power stations.

B. STANDARDS EVALUATION CRITERIA

This section considers the terms of reference of the Atomic Energy Control Board, plus the specific application areas for software within the nuclear power industry, and determines criteria for evaluating the various SQA standards and practices considered in later sections of this report.

1. A fundamental principle of AECB operation [Refs. 1 and 2] is that primary responsibility for nuclear safety and environmental protection resides with the licensee. Hence all proposed SQA procedures and practices should be considered as guidelines and do not supercede or remove the licensee's reponsibility for the ultimate quality of the software. It should be noted that currently some Standards are mandatory but this practice in no way affects the Licensee's primary responsibility.

2. "The credibility of the nuclear regulatory process depends not only upon its technical correctness and practicability, but also upon acceptance by the public-at-large of its perceived effectiveness and efficiency" [Ref 1, page 2] Accordingly, the selected SQA elements should constitute a blend of industry standard, state-of-the art SQA procedures, acceptable to both technical personnel and public interests as reflecting a solid, comprehensive, sensible but 'tight' approach to the safety-related aspects of the nuclear power industry.

3. The primary interest of the AECB in the safety and reliability (as opposed to development cost, ease of maintenance or change etc.) places the SQA emphasis on:

   a. correct, complete implementation of the functional requirements
   b. accuracy and timeliness of computer-based system response
   c. completeness of developmental and in-service testing

   This primary interest should be reflected in the selected SQA elements.

4. The regulatory process and its criteria and principles should be as concise as possible, clearly stated, and as understandable as possible [Ref. 1, page 2]. Accordingly, the selected SQA elements should be as clear, complete and definitive as possible.

5. The regulatory process used by the AECB for the past 35 years, is oriented toward the granting of a "Licence to Operate" subsequent to confidence gained via submissions and visibility during the development and commissioning phases. The selected SQA approach should therefore be compatible with this approach.

6. Quality of software comes largely from the professionalism and training of those engaged in its production. Conversely, it is difficult to ensure the quality of a product if those producing the product have no interest in producing good quality. Preference should therefore be given to those SQA elements and procedures that software developers are trained and familiar with, and that are highly regarded within the industry. Only where such practices are clearly inadequate or inapplicable to the nuclear power industry should 'proprietary' practices of the licensee or the AECB be used.

7. Whilst cost of development or the financial impact of practices is outside the terms of reference of the AECB, there is no doubt that it is easier for industry to respond where such matters are taken into account. Setting up a comprehensive SQA department and program is a costly exercise largely because of the influence on the management and development practices of the software producer. Hence it is difficult for a software development organization to correctly respond to more than one SQA program approach. Accordingly, and in compliance with criteria 6, every effort should be made to adopt one, complete, SQA program applicable throughout the software development industry. This criteria would rule out radical approaches to SQA requirements, but does not prevent the development of industry-specific SQA standards if such prove necessary.

8. The selected SQA elements should take recognition of the variable size, criticality and complexities of the range of software applications to be considered, and via customization, the SQA procedures should be applicable and amenable to both large and small, long and short-term projects. In short, the program should be applicable to a full range of software projects and all life cycle phases.

9. Studies indicate, and practical experience proves, that it is difficult to remove all bugs from complex software systems. This fact is reflected in the 'replication and interlocks' approach to critical systems within the

industry (e.g. nuclear power, mission-critical military systems, air traffic control). Experience also shows that the occurence of software bugs decreases approximately exponentially with system life. Therefore the software life-cycle implicit within the selected SQA elements should attempt to:

a) push the start of in-service operation as far as possible downstream, and
b) maximize the running experience prior to in-service operation.

10. The selected SQA elements should be as free as possible of individual or personal bias or preferences. Rather they should reflect industry-wide good practice and scientifically developed principles.

11. Software engineering is a dynamic discipline, and considerable development in SQA and management practices in particular has been evident in recent years. Accordingly the SQA elements selected should be, as far as possible, flexible and accomodating to future improvements, whilst eliminating the failing of the present and past.

12. The majority of AECB licensee's are extensively involved with hardware quality assurance, and there are specific QA standards already in place for the nuclear industry (CSA N286 series, see later section C parts 7 and 8). Hence the selected SQA elements should be capable of implementation within the general QA approach already adopted within the industry, whilst also catering for the unique requirements of the software life cycle.

13. It must be recognized that customer/developer and licensee/AECB relations can be by nature, a conflicting one. The time to negotiate is when both parties have something to gain. Hence the selected SQA plan should require the variable or interpretive items to be clarified at the start of a project, rather than as each requirement arises in the development cycle.

14. An important safety aspect of the nuclear industry is the complete recording, reporting and analysis of untoward events that occur during commissioning and operation. SQA procedures should ensure full visibility into such incidents where software is involved.

C. THE STANDARDS AND THEIR ASSESSMENT

This section of the report analyzes a number of available software standards current within the software industry. An attempt has been made to ensure the inclusion of standards from a wide range of applicable areas, differing standards bodies, commercial software development companies and geographical locations. In order to ensure uniformity of assessment, the various documents are analyzed under a number of headings, as follows:

1. Description of the Standard:
   This section consists of a brief description of the standard's producer and the major characteristics of the analyzed document.

2. Areas of Applicability:
   This section is intended to detail the target area or 'client' of the standard, and hence to clarify the applicability of the document to other environments.

3. Features and Exceptional Points:
   This section starts the analysis process by extracting the major points of the standard. A brief description of the areas covered is presented, as well as any exceptional features of particular interest for later section D - Extraction of Major Required Elements.

4. Strengths and Weaknesses:
   The particular good and bad points of the analyzed document are presented here, with special reference to any inherent limitations or failings. This section is the major analysis section for each of the examined standard documents.

5. Software Development Impact:
   This section makes brief comment on the impact the standard is likely to have on the software life cycle, particularly the development phase costs and schedules.

1. CSA Preliminary Standard Q396.1 - 1982 Software Quality Assurance Program, Part 1

1.1 Description of the Standard

This Canadian standard specifies the requirements of a software quality assurance program to be implemented and followed by a software developer/contractor in order to ensure the delivery of a quality product to a client. As such, the standard specifies the generic requirements of a software quality assurance program.

1.2 Areas of Applicability

This standard is applicable to a range of software applications (e.g. both real-time and data processing). It is written to be particularly applicable to the software customer and developer environment, i.e. it is intended to be a standard specified by a software purchaser and applicable to the activity of a software developer.

1.3 Features and Exceptional Points

The standard is designed to be compatible with the CSA Z299 Quality Assurance program standards, but applies to the software (as opposed to the hardware) production cycle.

1.4 Strengths and Weaknesses

a. The standard contains a fairly complete description of the requirements of a software quality assurance program.

b. The standard is a generic document. As such, it is weak in the area of detailed procedures and practices to be followed, compared with other available standards (e.g. DOD-STD-2167 Military Standard). It is understood that the CSA is proposing to provide some details in an accompanying guide.

c. The standard is somewhat limited by its need to address the customer/developer interface, rather than the more general "what is necessary for quality in software" question.

d. The need for reviews (functional requirements, preliminary and critical design etc.) while addressed in sections 5.1 and 5.2, needs clarification and expansion, as peer review is a very important part of ensuring (and teaching!) quality in software.

e. Whilst the generic need for many constituent parts of a full SQA program are called up, the majority of requirements are merely listed rather than completely described. Hence there is a possibility of interpretation differences with regard to the implementation of many items.

1.5 Software Development Impact

It is felt that this standard is well within the abilities of any reputable software developer, and indeed probably constitutes a minimum requirements list for a software quality assurance program.

2. BS 5887:1980 Testing of Computer-Based Systems

2.1 Description of the Standard

This is a British Standards Institution Code of Practice, developed to assist purchasing and software development organizations to specify, develop, document and evaluate general software testing strategies.

2.2 Areas of Applicability

The checklists, strategies and approaches outlined in this document are applicable to all software requiring any sort of testing. It is also applicable or useful to any organization involved in the evaluation of proposed testing or testing strategies i.e. a purchaser of custom software being developed by a software supplier.

2.3 Features and Exceptional Points

This code of practice considers seven areas of software testing:

Test Strategy - An organizational document detailing available aids, resources, techniques and methods.

Test Procedure - A complete list of all required procedures for each of the available test methods and techniques within the organization test strategy.

Test Criteria - A project-dependent document detailing the criteria for deciding the applicability of the test methods to a particular project.

Test Plan - A structured list of the tests to be performed on the system, detailing relationships between each test and the requirements.

Test Specification - The detailed procedures and instructions for test execution on a project.

Test Summary - A record of test results and the degree of conformity of the system with the original design criteria.

System Monitoring - Considers the requirements for in-service monitoring of system operation as regards the continued meeting of design goals.

2.4 Strengths and Weaknesses

a. The document is written as a guideline intended to be used as the reader sees fit, rather than as a definitive standard that must be followed. As such, there is considerable room for interpretation and selection.

b. The document is strong with regard to the proposed documents and overall approach to testing. It provides for a fairly complete approach to the subject via the various included checklists and factors to be considered at each stage of a complete testing strategy and plan.

c. The document is weak in terms of definitive test methods or procedures to be followed, i.e. in low level detail or mandated procedures to be applied.

d. The document takes recognition of the complexity, size and 'vital-ness' of the system to be tested, and provides guidelines as to test items and documents to be produced in such circumstances.

e. In the author's view, the document has a failing in that it does not emphasize the importance of unit or item testing. That is, testing of each individual module or program of a large system. Whilst mention is made of this item (4.3 item h), in view of the importance of unit testing in developing a correct system, it is not highlighted enough.

f. The document is also weak in considering testing responsibility and reporting aspects. Mention only is made of a final test summary. No mention is made of the roles of the original software developers, the software QA function, a separate test group etc. The position of testing within the configuration management function is also not considered.

g. No mention is made of the need for 'repeatability' of tests, such that they can be identically rerun subsequent to fixes and changes.

In conclusion, this document spells out a fairly complete

approach to the subject of software testing, without providing mandated or definitive methods, procedures or test plans to be followed. As such, it is ideal for a developer/licensee to address early in a project when clarifying his proposed testing approach. The document is weak in terms of addressing testing authority, responsibility, and reporting/recording aspects.

2.5 Software Development Impact

The adoption of comprehensive testing procedures definitely has cost and schedule impact on the development aspects of a system. It is likely however, to save time in the integration and commissioning phases of a project. The smaller the project is, the greater the relative cost impact. For a larger project however, where development staff are likely to be of 'average' capability, it is probable that additional time spent testing at the development stage will be more than recovered during integration and commissioning. Indeed for very large projects, it is unlikely that integration and commissioning will ever finish unless comprehensive testing is done earlier.

3. BS 6238:1982 Performance Monitoring of Computer-Based Systems

3.1 Description of the Standard

This is a British Standards Institution Code of Practice concerned with monitoring the performance of a computer-based system in service.

3.2 Areas of Applicability

This document is applicable to all computer-based systems in which there is an ongoing interest in the maintenance of the speed, performance and response times of the system. This is not currently considered as a software quality assurance item, but should be of interest to the AECB with regard to the response times of reactor control and safety systems.

3.3 Features and Exceptional Points

This code of practice addresses the following items:

Likely causes of change in the system being monitored, in the areas of applications software, operating and support software, hardware configuration changes and ageing, and system inputs and usage.

Monitoring techniques available to control the above change areas. These include routine system records, hardware and software performance monitors, in-service testing, system audits and user surveys.

3.4 Strengths and Weaknesses

a. This code of practice is rather more of a checklist or description of items to be considered. As such, it is useful if considering the subject of system performance monitoring prior to the fact, but the document is too open to interpretation etc. for it to be used as a definitive, mandated standard.

b. The subject of on-going performance monitoring is not normally considered as a software quality control item, as the items causing performance variations during the operational lifetime of a system are more hardware or system usage related. The software changes that may

impact performance are considered under Configuration Control.

3.5 Software Development Impact

The requirement for software monitors, or the need for in-service testing features, may require additional software to be produced thus having a software development impact.

4. BS 5515:1984 Documentation of Computer-Based Systems

4.1 Description of the Standard

This British Standard Code of Practice provides guidance on the required documentation of a computer-based system.

4.2 Areas of Applicability

The document is structured so that it relates to each of the project life cycle stages. The aim of the standard is to establish a basic framework of documentation for a computer-based system. The documentation called up by the standard is applicable to any computer-based system, though obviously the 'documents' may be minimal for a smaller programming project.

4.3 Features and Exceptional Points

This standard first defines the purposes and principles of documentation, and then defines the life cycle of a computer-based system. The defined nine stages, with the called-up documentation are as follows:

1. User Requirement stage:
   User Requirement, Feasibility Study Report

2. Feasibility Study stage:
   Feasibility Study Report, User Specification Request.

3. User Specification stage:
   User Specification, System Specification Request.

4. System Specification stage:
   System Spec., Implementation Plan, Procurement Spec., Systems Development Request.

5. System Development stage:
   System Manual, User Manual, Operations Manual, Data Manual, Program Manual.

6. Acceptance stage:
   Acceptance Certificate, Acceptance Test Request.

7. Handover stage:
   Post-implementation Review Report.

8. In-service System Support stage:
   Pro Forma's, Authorization for Disposal.

9. Disposal stage:
   Disposal Certificate.

4.4 Strengths and Weaknesses

a. The standard presents a logical, structured arrangement of related documentation throughout the life cycle of a computer-based system, however the steps and stages plus the called-up documents do not agree with or conform to that currently prevalant in the North American software industry.

b. The standard makes no clear distinction between, or even allowances for, a customer/supplier, user/developer or licensee/licensor arrangement. This affects the allocation of contents to the various documents plus the Acceptance Criteria arrangements etc.

c. The standard is confusing on the System Specification and System Development stages. Many development items appear to be called up under the System Specification stage (e.g. program and file details).

d. The standard should take more recognition of the fact that systems vary, and provide suggested checklists of document contents at each stage, rather than what appears to be mandated items. Possibly a few words should be added concerning mandated items (e.g. program descriptions and file details) in that they must appear in either the System Specification or the System Manual.

e. The standard is weak in the System Development stage. There appears to be no definitive requirement to document and describe the design and the individual program details (to any level!) before coding of each module commences.

f. There is an Acceptance stage defined but no specific call up or mention of the various tests or test documents that Acceptance is normally dependent on.

g. The standard refers to a feasibility study stage, a handover stage, and a disposal stage, with no mention or allowance that these may be optional stages. Not all projects have a specific feasibility stage - many proceed direct from User Requirements to User Specification, in fact in many projects the Requirements are the Specification.

h. The appendix A, Cross-referenced Information Checklist is a good aid to ensuring that all required documentation items are included; or if they are omitted, that there has been a definite decision to do so. This is a good method of allowing options and variability in standards, whilst forcing a licensee to specifically answer why he is not supplying all referenced items.

i. The standard has the strong point in that it does refer to a software life cycle, and ties documentation to the various stages. The feeling of this author is that many other SQA items (e.g. Reviews and Audits, Configuration Management, and Testing) are related to the software life cycle, and hence it may have been better if this had been specified in the higher-tier standard.

In conclusion this Code of Practice presents a coherent set of documentation requirements that do not conform to current North American practice, and the standard is weak in a number of areas.

4.5 Software Development Impact

This standard should have minimal impact on software development as it is assumed that a developer is providing most of these documentation items already, albeit in another form. Certainly this standard does not refer to any new, extensive documentation items.

5. BS 6488:1984 Configuration Management of Computer-Based Systems

5.1 Description of the Standard

This British Standards Institution Code of Practice gives recommendations for the establishment of configuration management procedures for computer-based systems.

5.2 Areas of Applicability

This configuration management standard is applicable to any organization involved with the handling of changeable software and hardware, that is, systems either being developed, or being enhanced and upgraded.

5.3 Features and Exceptional Points

This standards document addresses the basic principles and major items of a configuration management plan. These consist of the need for items to be tracked and recorded (Configured) in baselines, and to be listed in related baseline configuration indexes. The various related items of documentation, status accounting, periodic audits and reviews are also briefly addressed.

5.4 Strengths and Weaknesses

a. The document is a reasonably good overview of the basics of software configuration management. It provides much justification and background for the subject but little definitive detail.

b. The document admits that what is considered as a configurable item (an individual source module, a complete system etc.) is crucial to the effectiveness of configuration management. The document does not however provide enough guidelines as to the nature of a configurable item. It also does not address in sufficient detail, the important issue of when an item should be configured (during design stage, when first coded, on initial delivery to the customer etc.).

c. The document does not make clear the need for version or revision numbers in each release of a configured item. It does however address the need for recording the identity

of each configured item in each baseline release of the system.

d. The document makes little mention of the related items of a system or configuration library for software and documentation. There should be, as a minimum, some recognition of the need for centralized control and storage of configured items.

5.5 Software Development Impact

Configuration Management has some impact on the cost of software development, but the cost of not having such a scheme and then losing track of vital components is much greater.

6. Department of National Defence System Development Methodology (DND SDM)

6.1 Description of the Standard

This document is not a quality assurance document, but is a project management plan. That is, it mandates the life cycle phases, the various tasks undertaken at each phase, and the documents, forms and products produced by each phase of the project.

6.2 Areas of Applicability

The project life cycle procedures described in DND SDM apply to all National Defense Headquarters (NDHQ) controlled Automatic Data Processing (ADP) projects that fall within certain DND policies. The publication also contains official recommended guidelines for all other DND ADP projects.

The DND SDM contains material based in part on SDM/70, a proprietary product of AGS Management Systems Inc. All materials included are therefore commercial confidential. The publication is for use only within the Department of National Defence and by DND personnel. Express written permission must be received from ND HQ/DG Man S/DISP in order for non-DND personnel to gain access.

6.3 Features and Exceptional Points

DND SDM currently consists of seven volumes of detailed project management tasks, procedures, forms, and guidelines. Volume 1 contains introductory material and summary information related to all nine identified phases of the defined project life cycle. The remaining 6 volumes provide the detailed tasks, procedures and forms for each of the nine phases.

6.4 Strengths and Weaknesses

a. One of the major limitations of this plan is that it is so large and detailed that it is a major task merely reading it! Indeed it is intended that this plan be used only on fairly large data processing projects. Hence, while there is much software engineering content in this set of documents, it is perhaps too unwieldy for most real-time projects.

6.5 Software Development Impact

This standard has a fairly considerable effect on the software development effort in that it mandates the management and monitoring approach. As such it is likely to conflict with existing software engineering standards, hence it is envisaged that only this standard would be used for project management on a project.

7. CAN3-N286.1-84 Procurement Quality Assurance for Nuclear Power Plants

7.1 Description of the Standard

This Canadian Standard specifies the quality assurance program requirements to be met by a nuclear power station owner, and his designated participants, in the procurement of safety-related equipment, systems and structures. It is the second edition of CSA N286.1 previously issued in May 1979.

7.2 Areas of Applicability

The standard is applicable to the procurement of items or services for nuclear power plants. It specifies the quality assurance program requirements to be met by the owner and his designated participants in the procurement of safety-related equipment, items or services. Whilst not specific to software quality assurance, obviously any procured software items should, in some fashion, be covered by this standard.

7.3 Features and Exceptional Points

The N286 standard series are administrative in nature rather than technical. The standard therefore initially specifies the management functions of the owner and participants. That is, the responsibilities, policies, organization, authority, and qualifications of the personnel involved in safety-related systems purchasing. Also specified are the contents of the quality assurance program manual, the need for periodic program reviews, the relevant interface controls, feedback, and correlation to other standards.

The standard then calls up the required functions to be performed during the procurement process, including document preparation, the incorporation of QA requirements in bid documents, procedures for evaluation and selection of suppliers, and bid evaluation and award measures. The standard specifies that N286, CSA Z299, or other QA standards are to be incorporated as appropriate in the procurement documents.

The verification, audit and corrective action functions necessary to a complete quality assurance program for the procurement of safety-related items within the nuclear power industry are then specified. Finally the subject of document

and records control and retention is addressed.

7.4 Strengths and Weaknesses

a. The standard is a very good, high level specification of a QA program to be applied to the purchase of items and services.

b. The standard has a fairly good preface which provides background information, and details the relationship with other standards etc.

c. It is fairly general and non-specific with regard to the items and services to be procured, specifying only that they be safety-related equipment, systems and structures as identified by the owner. With the addition of relevant lower-level SQA standards to section 4.2.3, the Standard could be applicable to software procurement.

d. Section 1, Scope, makes no mention of the applicability of the standard to safety-related 'services'; possibly section 1.4 should read '...procurement of safety-related equipment, systems, structures, and services.' It is noticed that the previous CSA N286.1 - 1979 standard had the key word "all" at the beginning of this phrase! Mention is however made of services in section 3.1.4.

e. The standard is applicable to the Owner and "participants designated by the owner". A participant however, is defined as someone to whom the N286 standards apply (see Section 2. Definitions)! Not only does there seem to be a circular type of reference here, but little information is provided as to who the owner should or should not so designate. Similarly with regard to the section 1.4 usage of "safety-related equipment ... as identified by the owner". There seems to be the possibility that the owner may select when the standard should be applied, independent from the safety aspects.

f. The standard has been written specifically for the Canadian nuclear power industry with regard to the owner of a nuclear power plant, and his various participants. As such, interpretation difficulties would arise if the standard was called up in other circumstances, e.g. by a commercial software system procurer on his software supplier and the supplier's subcontractors.

g. One of the major weaknesses of the standard, in this author's opinion, is that it is unclear, and no guidelines are given, as to how far down the supply chain the standard should be applied. It is simplistic to say 'all the way' as a nuclear power plant involves

potentially many levels of suppliers, from major component suppliers (building, reactor, shutdown systems etc.) right down to steel fabrication, nuts and bolts, wire and insulation on wire etc.

h. There is no definition of 'safety-related equipment, systems and structures', nor any guidelines etc. supplied as to what should be covered by this phrase.

7.5 Software Development Impact

It is considered that the standard is applicable to the procurement of software or software services, and that the activities called up would be part of the normal procurement procedures involved in the provision of such items. However some potential software suppliers (e.g. University departments, off-the-shelf software suppliers, software personnel agencies) may have difficulty meeting the requirements within their current organizations.

8. CSA N286.2-1979 Preliminary Standard. Design Quality Assurance for Nuclear power plants.

8.1 Description of the Standard

This Canadian preliminary standard specifies the requirements for a quality assurance program to cover the activities related to the design phase of a nuclear power plant.

8.2 Areas of Applicability

This standard is applicable to the design of safety-related structures, systems and components. It may also be applied, in whole or in part, to other items of the plant.

The standard is applicable to all phases of the design work, including concept studies, preliminary design, detailed design, and the production of design technical documentation for licencing, purchasing, constructing, installing, testing, commissioning, and operation activities.

It is not specifically applicable to software design activities, but certainly such activities could be covered by this standard.

8.3 Features and Exceptional Points

As an administrative N286 standard, this document follows the general format of others in the series. The management functions of the owner, and other participating design organizations, required to assure quality in the design, are addressed first. The various functions to be performed during the design process are described in the next section. Finally the verification and audit procedures are addressed, followed by the design records to be established and maintained. The standard also includes a number of non-mandatory appendices containing suggested lists of items and topics for use during various stages of the design assurance process.

8.4 Strengths and Weaknesses

a. The standard includes a short but concise preface that explains the relationship of this standard to other relevant standards, and also explains the purpose of this

standard.

b. The standard, as stated in section 1.2, Applicability, applies to the design of safety-related structures, systems and components. Software is such a key part of many safety-related systems that the feeling of this author is that specific mention should be made of it at this point.

c. Section 2.1 of the standard contains a good definition of the key phrase "safety-related equipment and systems".

d. Section 5, Design Verification, contains good requirements for the completed design to be subjected to verification. It should however be stated that the design, or each part of the design, be reviewed by technical personnel familiar with the requirements, preferably qualified to have undertaken the design themselves, yet independent from the actual designers (e.g. the peer-related critical design review process)

e. An important aspect of the design process is to consider alternate designs and approaches. These alternate designs plus, most importantly, the reasons for their rejection and modification, should be documented and available for verification of the completeness and technical accuracy of the design process.

f. Section 3.2, Policy, is good that it specifies that a written policy should be issued by management, and formally acknowledged by the relevant management levels. This author believes that it is very important that such policy statements be disseminated amongst all owner staff, in order that the organization commitment should be adopted by those actually tasked with producing the product.

8.5 Software Development Impact

The application of this standard to the design process of a software product should have purely a beneficial effect on the software development. It is difficult to envisage a reasonably sized software development effort where such design activities as called up in the standard are not, in some form, performed.

9. FAA-STD-018 May 1977, Federal Aviation Administration, Computer Software Quality Program Requirements

9.1 Description of the Standard

This standard from the FAA Department of Transportation establishes the minimum requirements for a computer software quality program to be established and maintained by a contractor under FAA contract for the furnishing of computer software and services.

9.2 Areas of Applicability

This standard applies to the acquisition of computer software programs and related documentation when referenced in an FAA contract or specification. Note that related standards FAA-STD-013 Quality Control Program Requirements, and FAA-STD-016 Quality Control System Requirements, form part of this standard, and are therefore applicable to the extent specified in each particular FAA contract.

9.3 Features and Exceptional Points

This short, five page document specifies the requirements for a Computer Software Quality Program Plan (CSQPP). The requirements called up, whilst being fairly brief, do seem to include most topics that should be included in such plans. The specific items to be addressed by the CSQPP are as follows:

3.1.1 Organization
3.1.2 Development Process Flow
3.1.3 Standards of Software Quality
3.1.4 Computer Software Work Instructions
3.1.5 Identification of Process Status
3.1.6 Configuration Management
3.1.7 Design Review
3.1.8 Corrective Actions
3.1.9 Test Controls
3.1.10 Tools, Techniques and Methodologies
3.1.11 Subcontractor Controls
3.1.12 Records
3.1.13 Audits

See the further comments below for the exceptional points of this standard.

9.4 Strengths and Weaknesses

a. The standard seems to define a fairly complete contents list for a software quality assurance plan. Note that this standard requires a project-specific document. Sufficient time was not available to examine the referenced FAA standards that probably describe the general organization-wide SQA program requirements.

b. Many items in the standard refer to the contract for detailed QA requirements, e.g. "Design reviews shall be accomplished in accordance with contractual requirements". This seems to be pointless as the contract would always be enforced, and the purpose of calling up standards in a contract is precisely to avoid putting QA etc. requirements in detail in every contract.

c. This standard has a number of interesting called up items not encountered by this author in similar standards such as CSA Q396.1, IEEE 730, and DOD-STD-2167. For example:

   1. Section 3.1.2, Development process flow, calls up flow charts or narrative to describe the "sequence of operations" by which the software development is to be undertaken. This would seem to be a useful way of identifying organization, responsibilities, and activities.

   2. Section 3.1.3, Standards of software quality, calls up specific acceptance criteria for the various design, development, and evaluation activities. Perhaps this is merely another way of describing internal company standards, but it is a good feature that the document mentions the need for evaluation criteria at each project stage.

   3. Section 3.1.4, Computer software work instructions, calls for procedures to cover the generation of "work instructions" for developing software. This would appear to be a good method of instructing and controlling the workings of the various parts of the developer's organization. It also gives the customer visibility into these workings.

d. The standard does not call for any functional requirements documentation or reviews of such. It also calls for only one design review, rather than for preliminary and critical reviews. The standard also does not call for submission of a detailed test document or acceptance document to the customer, though the

contractor can of course commit to do this in his response to section 3.1.9 Test controls.

e. The section 3.1.11 Subcontractor Controls, seems weak in not specifically requiring a similar SQA program commitment from subcontractors.

In short, this standard calls up a fairly complete SQA plan, but leaves much to the original contract, and provides little guidance to a contractor as to the details he should include in the plan. This author feels that a new supplier to the FAA may have considerable negotiation and change etc. to do on his plan prior to acceptance by the customer. The avoidance of this activity of course, is one reason for the creation of detailed standards.

9.5 Software Development Impact

The called up items in this plan should have little development impact on a developer already involved in critical-software production, as it is felt that many of the procedures would already be in place. However the implied project management and control, via work instructions etc. and the terminology, may well be different from that currently being used by a contractor involved in, say, MIL-STD-1679A production. This may have some software development impact.

10. DGMEM SOFTWARE STANDARDS, 11-1979, Produced by The Canadian Forces

10.1 Description of the Standard

This document, produced by the Director General Maritime Engineering and Maintenance (DGMEM) department within DND NDHQ, consists of four volumes.They contain guidelines for the development and maintenance of computer software and its associated documentation.

10.2 Areas of Applicability

The document is not intended as a standard, or to replace standards. Rather it is an addition, in that it is designed to complement other DND documents (CFP 113) related to military systems life cycle management, by providing guidance on software topics. It is germane to this report in that there is much to be learned by the standards-writer from the sensible, practical guidelines included.

10.3 Features and Exceptional Points

The document consists of four volumes as follows:

Vol 1.	The Software Life Cycle	18 pages
Vol 2.	Guidelines	21 pages + 8 annex pages
Vol 3.	Documentation	5 pages + 25 annex pages
Vol 4.	Reviews and Audits	14 pages + 4 annex pages

Volume 1 describes a seven stage software life cycle, from project definition stage through to an in-service stage. This is intended to form a framework to be used for planning, scheduling of reviews and audits, control of documentation, and project progress monitoring. Volume 1 is referenced by the later volumes, and therefore defines each stage in terms of its objectives, activities, products, and related reviews/audits.

Volume 2 provides point-type guidelines for each of the software production stages of the life cycle, plus guidelines for particular QA-type activities. Also provided is a "how to live with guidelines" section. It should be noted that all these volumes are of interest to both the customer (DND project office) and the developer (DND contractor). An annex to the volume provides guidelines for the design and coding of software modules.

Volume 3 contains specifications for the format and content of each documentation item of a software project. Nine documentation items were called up in the volume 1 life cycle description, and this volume provides an overview of each plus detailed guidelines for each one in the 9 annexes.

Finally volume 4 provides guidelines for the minimum four reviews and two audits called up by volume 1. Annex A of this volume contains a Software Quality Assurance Program Specification.

10.4 Strengths and Weaknesses

a. The definition and explanation of the project life cycle, located up front in volume 1, enables the whole of the remaining volumes to use this as a framework in which to define documentation, reviews, evaluation criteria etc. Note also that the defined project life cycle accounts for both hardware and software, in that the software-dependent portions of the project are separated out at the end of the system design phase. This approach is of particular value for nuclear industry safety-critical systems, as they similarly consist of combined hardware and software.

b. Volume 1 specifies in section 3.1, Assumptions used in the Life Cycle Definition, the criteria to be used in deciding the applicability of DGMEM guidelines to projects (e.g. more than two people will be involved in the development). This is a good feature.

c. Section 12 of volume 1, Variations in Project Characteristics, provides some guidance on modifying the defined project life cycle to handle special cases. This is a good way to respond to the objection that any defined life cycle is not universal.

d. The document is, of course, open to the criticism that it is not intended as a standard, and hence is subject to different interpretations etc. Note that section 1.0 of volume 2, Scope, states "This document contains a series of guidelines which offer directions for those aspects of a software project where standards would be ineffective or inappropriate".

e. Volume 2 does not lend itself to ready transformation into a standard. There is however, much wisdom in the volume to be considered when drawing up the standards covering different project stages. There is also much food for thought for AECB

staff called upon to monitor applicant's software development efforts.

f. Although volume 4 contains an SQA program specification annex, no reference to this annex could be found in the rest of the document, and the subject of SQA was conspicuously absent. The conclusion is therefore that this document is intended to be a contract management document to be used in conjunction with other SQA requirements (e.g. MIL-STD-1679A).

g. The project documentation details are given in the volume 3 annexes. Generally the called up contents are the minimum that should be required for safety-critical software. In particular, the testing documents are very weak.

h. Volume 4 provides details of the various proposed reviews and audits. It is good that the topic of evaluation criteria for each review is addressed, but it is felt that these criteria could be expanded upon. Also not described or considered is the desirability of an "adversarial" arrangement in reviews, i.e. the requirements writers do not do the software design but do take part in the software design review; the software designers do not do coding but do review/audit the code tests etc.

i. Annex A to volume 4, Software Quality Assurance Program Specification prompts the following comments:

   1. This annex seems to confuse a SQA program (conglomeration of people, procedures, records etc.) with an SQA plan (a document that describes, possibly, an SQA program). This author feels that a contractor should be required to have an SQA program, that is described and submitted for customer approval in an SQA plan document.

   2. The annex states that the purpose of an SQA program is to ensure that delivered software complies with the program requirements. If this is the only purpose of an SQA program, this purpose could be satisfied via testing and reviews. This author feels that the attainment of "quality" in the delivered product, including implementation of all defined requirements, is the true purpose of an SQA program. Accordingly, more is needed than is asked for in this annex. For example, no details are given in section 3.2.7, Software Documentation, of the required software coding standards; in section 3.2.8, Reviews and Audits, no specifics are given as when and what should be reviewed.

3. The annex is confusing in considering the subjects of configuration management, corrective action, and library controls as separate topics. Configuration management consists of library controls, corrective action procedures etc.

10.5 Software Development Impact

These DGMEM documents do not consist of standards to be followed, so much as topics to be considered in software development. The life cycle described and the documents to be produced would impact development costs and/or schedules if not previously followed, but would not imply the overhead associated with more rigorous military standards e.g. MIL-STD-1679A or DOD-STD-2167.

11. ANSI/IEEE Std 730-1984. Software Quality Assurance Plans

11.1 Description of the Standard

This American standard is designed to assist in the preparation and content of software quality assurance plans, and also provides a standard against which such plans can be assessed.

11.2 Areas of Applicability

This standard is directed toward the development and maintenance of critical software, that is, where failure could impact negatively on safety or cause large financial or social losses. There are three groups with an interest in this standard, as follows:

- The software user who has a need for the product. Due to his time and financial investment, he needs to obtain a reasonable degree of confidence that the software is acquiring the desired quality during the development process.
- The software developer needs an established standard against which to plan, organize, and be measured.
- The public who have certain legal rights with regard to system usage. They require the confidence that user and developer have acted in a reasonable, prudent, and professional manner in regards to the system development.

11.3 Features and Exceptional Points

The purpose of this standard is to provide uniform, minimum acceptable requirements for preparation and content of software quality assurance plans. As such the document is structured around the proposed table of contents for the SQA plan. Specifically it contains the following sections:

3.1 Purpose: delineates the specific purpose and scope of the plan, and lists the software items and uses covered by the plan

3.2 Reference Documents: a complete list of lower-tier documents called up by the plan

3.3 Management: describes the organization, tasks, and responsibilities of the various elements that influence the quality of the software.

3.4 Documentation: this section identifies five obligatory plus three optional documents to be produced during the development cycle. The standard briefly describes the documents and specifies that the SQA plan should also state the review or audit process for each document.

3.5 Standards, Practices and Conventions: This section of the SQA plan identifies the various lower level standards etc. to be used during software development.

3.6 Review and Audit: defines the technical and managerial reviews and audits to be undertaken, and states how they will be accomplished.

3.7 Software Configuration Management: describes the methods to be used to identify software items, plus control, implement, record and report changes to those items.

3.8 Problem Reporting and Corrective Action: procedures to accomplish the actions, plus relevant organizational responsibilities.

3.9 Tools, Techniques, and Methodologies:

3.10 Code Control: maintenance and storage of controlled software.

3.11 Media Control: protection and security of physical computer media.

3.12 Supplier Control: provisions for assuring that vendor-provided and subcontractor-developed software meets the technical requirements.

3.13 Records Collection, Maintenance and Retention:

11.4 Strengths and Weaknesses

a. The ANSI/IEEE pre-amble included in packaged copies of the IEEE software standards states that Std 730 is "sparse and thus provides a template for development of further standards, each expanding on a specific section of the document". This author would agree, but feels that such a sentiment should be included within the standard itself to warn prospective users of the need for amplifying and clarifying detail on many items.

b. The standard specifies the minimum contents and items to be addressed by a project document called the Software Quality Assurance Plan (SQAP). An SQA plan however, is not a stand-alone entity. It is a specific document produced as a

result of a developers committment to software quality, and as a result of experience and capability within a software quality assurance program. There is no mention of the overall program requirements under which an SQA plan may be called up (c.f. MIL-S-52779A, section 17 of this study).

c. The standard calls up many documents which, under various different names, are now common to many SQA programs and plans. However, there is no mention within the standard of the related topic of a software life cycle, and its various phases. The ANSI/IEEE Std 729-1983 (not formally reviewed in this study) does define the term "software life cycle", indeed it has a special section devoted to that topic. The feeling of this author is that a definition and understanding of a software life cycle is so much a part of the software engineering process on a given project, and so tied in to the documentation, review and test aspects, that it should be defined either within the standard itself, or within the SQA plan document.

d. Apart from the section 3.4.2.3, Software Verification and Validation Plan, the important subject of testing is little addressed in this standard. It is realized that there is a separate standard which addresses testing, but it is felt that this higher-tier standard should stress the importance of this topic (c.f. MIL-S-52779A section 3.2.8, and DOD-STD-2167 sections 5.4, 5.5 and 5.6). There should also be closer agreement between this standard and IEEE Std 829 with regards to testing approaches, nomenclature etc.

11.5 Software Development Impact

The standard itself has very little impact on the software development process. It is what the developer contracts to do within the SQA plan defined by the standard that is likely to impact a project. Obviously the actual software development impact is therefore very dependent on the lower tier standards of each called up item.

12. ANSI/IEEE Std 828-1983. Software Configuration Management Plans

12.1 Description of the Standard

This American standard provides minimum requirements for preparation and content of a Software Configuration Management (SCM) plan; i.e. a plan to identify developed software, control baselines, control changes to baselines, record and track status, and audit the software.

12.2 Areas of Applicability

This standard applies to the entire life cycle of a software system, particularly of critical software, and is normally of interest to software developers or software maintainers. However, as with all SQA standards, the software user and the general public have an interest due to the safety, financial or social losses involved in software failure.

It should be noted that there is a further standard, IEEE Std 1033-1985, Recommended Practice for Application of IEEE Std 828 to Nuclear Power Generating Stations.

12.3 Features and Exceptional Points

This standard describes the format and contents of a Software Configuration Management Plan (SCMP) document, with some initial guidelines as to by whom and how the plan is to be produced. The sections of the plan and their respective contents are as follows:

Introduction - The purpose and scope of the plan, plus definitions and related document references.

Management - The organizational entities involved in the plan and their various responsibilities. Methods for interface control, the SCMP plan implementation schedule, and the applicable policies, directives and procedures are also detailed.

SCM Activities - This section describes how the requirements for configuration identification, control, status accounting, and audits and reviews, will

be satisfied.

Tools, Techniques, and Methodologies used to support the SCM function on this project

Supplier Control - The impact of SCM requirements on subcontractors, and vice versa.

Records Collection - SCM documentation retention, safeguards, and retention period

12.4 Strengths and Weaknesses

a. The standard is a fairly good and complete description of the contents of a Software Configuration Management Plan. There are, however, some deficiencies as described below.

b. There is little in the way of guidelines or descriptive material on the uses or purpose of an SCMP. For example, no mention is made of the need to have the plan in place prior to project software or documentation production.

c. There is no mention or recognition of the developer/client relationship, hence no mention of submission to or approval of the plan by the client.

d. The standard makes reference to a Configuration Control Board (CCB), and a called-up document section is intended to cover the organizational relationship of the plan to the CCB. But there is no description of the CCB and it is not clear if this board is part of the SCMP or is an outside organization.

e. The standard should take some recognition of the configuring needs of source modules versus binary modules (relocatables), complete subsystems (runnable task or process images), and complete systems. For example, is it required that a system version number be updated if a single source module is changed ?

f. Most real configuration management organizations are tasked with controlling documentation as well as software. Whilst it is realized that this standard addresses only the software issue, some recognition should be given to the need for, integration with, and relations with, the similar documentation configuration problem.

g. The standard does call up details of the required level of testing necessary for acceptance of an item into configuration management. It does not however, mention the need for retesting or changed tests for updated or replaced configured items.

h. A good point is that the standard does call up the need to apply configuration management to the related software development, test, and support tools.

12.5 Software Development Impact

It is felt that the essence of a configuration management system should be in place in all organizations involved in the production or maintenance of software. Hence the major adverse impact of this standard is in the necessity of formally documenting and tightening up on in-place procedures.

13. ANSI/IEEE Std 829-1983. Software Test Documentation

13.1 Description of the Standard

This American standard describes a set of basic test documents which are associated with the dynamic aspects of software testing (that is, the execution of procedures and code).

13.2 Areas of Applicability

The standard may be applied to commercial, scientific or military software which runs on any digital computer. Applicability is not restricted by size, complexity or criticality of the software.

The standard should be of interest to software users, procurement personnel, to development, test and maintenance personnel, to operations and acquisition support managers, to SQA personnel and auditors, and to those involved in legal aspects of software.

13.3 Features and Exceptional Points

The documents outlined in this standard cover test planning, test specification, and test reporting.

The test plan prescribes the scope, approach, resources, schedule, risks and responsibilities of the testing activity. It identifies the items and features to be tested, the tasks to be performed, and the pass/fail suspend/resume criteria.

The test specification is covered by three documents. The test design specification refines the test approach and identifies the features to be tested by this design and its associated tests. The test case specification documents the actual input and expected output values, plus the resources and procedures for each test case. The test procedure specification documents the steps for executing a set of test cases or, more generally, the steps used to analyze a software item.

Test reporting is covered by four documents. A test-item transmittal report provides the information for, and is used to initiate actual testing of an item. The test log provides a chronological record of test execution, including results and

anomalous events etc. A test-incident report is used to document any test event requiring investigation or follow-up. Finally the test-summary report documents and provides the evaluation of the complete test results and activities.

13.4 Strengths and Weaknesses

a. The standard provides complete, detailed descriptions of a number of test-related documentation items, It also provides comprehensive examples via appendix A.

b. The standard does not provide much detail or guidance on document usage. There is an appendix to aid the customizing or initial introduction of the documentation set. There is no guidance as to what phases of the project life cycle the documents should be produced in, and who should produce, review and use the various documents.

c. There is no mention in the standard of the importance of verifying the correctness and accuracy of the testing techniques. There is no mention of the importance of the tests being repeatable, or of the desirability of automatic, verifiable hardcopy results output. There is no mention of the need for SQA procedures in the development of the test software itself.

d. It should be realized that verification of the correctness of a software system, whilst largely dependent on the correctness and completeness of the code testing, is also dependent on the correctness of the requirements and design phases (and in fact, is dependent on any earlier phases of the project). Verification and validation (via reviews, pseudo-code or program design language testing etc.) of these earlier phases of software systems production are not covered by this standard.

13.5 Software Development Impact

The amount of testing undertaken definitely impacts negatively on the schedule and thus the apparent cost of a software project, particularly in the case of full system testing, which by its nature, must be undertaken after completion of development. The industry is divided as to whether comprehensive module or unit testing adds to the cost of the development phase or not. What is clear however, is that failure to undertake adequate testing, particularly in the earlier stages of development, can result in major problems

further downstream, and add to the overall life cycle cost. Current industry philosophy is therefore "the more testing, the better!".

14. ANSI/IEEE 830-1984. Guide to Software Requirements Specification

14.1 Description of the Standard

This IEEE guide (not a standard) describes topics to be considered and a proposed table of contents etc. for a Software Requirements Specification (SRS).

14.2 Areas of Applicability

This guide is applicable in any situation where a client, user, or a software developer, requires to clearly, completely, and unambiguously specify the features and functions of a software-based system.

This would be a requirement of most systems (possibly prototype or research-type systems might be exceptions), and has the following benefits:

1. Establishes the basis for client/supplier or applicant/licensee agreements.
2. Serves as input to the development effort, reduces such efforts, and provides a basis for the development cost and schedule estimates.
3. Provides a baseline for validation and verification at the system test stage.
4. Serves as a basis for future portability and enhancement efforts.

The foreword of this IEEE standard provides further information on the areas of applicability of the SRS.

14.3 Features and Exceptional Points

This guide consists of basically three parts. The first part is background information for writing a good software requirements specification. That is, characteristics to aim for in the document, and topics to be considered prior to the writing effort.

The second part of the guide consists of topics to be considered with respect to the actual software requirement

items to be documented. For example, methods of expressing requirements, annotation of the requirements, and common pitfalls encountered in expressing requirements, are addressed.

The last part of the guide provides a suggested table of contents, plus defines the contents of each section of the SRS. Also provided are a number of alternate layouts for the main "functional requirements" section of the SRS.

14.4 Strengths and Weaknesses

a. This guide has the strong point that many items to be considered, and suggestions etc. are included. It is however, a guide rather than a standard, and it is therefore a question of judgement as to whether a given SRS "satisfies this standard". An alternate approach would be to create a standard specifying a software requirements specification (the last part of the current guide), and put the "guide" portions (parts one and two) in appendices.

b. The guide is very thorough when detailing the documentation etc. of each requirement and is, in general, a very impressive attempt to come to grips with a difficult and problem-prone part of software development.

c. The guide is good in emphasizing that software design items have no place in the SRS. That is, the SRS is intended to serve as input to the design process.

d. The guide suggests that both client and developer should produce the SRS. Whilst it is certainly true that both should have input and both should whole-heartedly agree with the finished document, the realities of separate locations and organizations etc. tend to preclude this common production. Hence section 4.4 of the guide should suggest alternate arrangements i.e. other party review and sign-off, joint requirements reviews etc.

14.5 Software Development Impact

This document has no impact on the software development effort directly, as this requirements analysis and documentation effort occurs before software development starts. Likely impact however is definitely beneficial in the long run as good requirement definition results in efficient, controlled software development.

15. A.P.E.O. Guidelines to Standards of Practice for the use of Computer Programs in Engineering

15.1 Description of the Standard

This document, produced by the Association of Professional Engineers of Ontario, is a set of guidelines (i.e. not a standard) regarding the use of computer programs used by professional engineers, in the practice of engineering.

15.2 Areas of Applicability

The guidelines are intended to apply to the use of computers and programs in the practice of engineering involving such areas as analysis, design, costing, scheduling, calculating, simulation etc . The guidelines are intended to indicate to both the professional engineer, and the public, the approved practices that an engineer should follow in the use of computers and programs in his/her profession.

15.3 Features and Exceptional Points

The document addresses the responsibilities of an engineer in the use of a computer program, and provides a broad list of "approved practices" items. In order to aid the engineer in the following of the above suggested practices, the document includes two additional parts.

Part II of the document details the characteristics of a good computer program. Such items as program organization, coding practice, self-checking, adaptability, expansion and alteration characteristics are described.

Part III addresses the issue of good documentation in order that an engineer may have some idea as to what is expected of a good program. Suggested contents and layouts of both engineering and software documentation, and of the users manual are provided.

15.4 Strengths and Weaknesses

a. This document is not intended to be an SQA standard, and hence does not spell out practices and procedures of great interest to us. It does however, contain useful information

with regard to the details required in computer program documentation, by a program user.

b. The document does mention briefly the idea of a computer program having self-checking capability, particularly with regard to computation results.

c. The document does have something to input to the Operational and Support sections of our review, in that it defines the needs of the computer program user. The document would be a help to a client in his initial drawing up of the computer program requirements for his software developer.

15.5 Software Development Impact

The document has no impact on the development effort of computer programs. It may well have some small additional impact on the task of producing user documentation, as it does suggest items to be included.

16. DoD-STD-2167, June 1985. Defense System Software Development

16.1 Description of the Standard

This United States Department of Defense standard specifies requirements for the development of military mission-critical computer system software.

16.2 Areas of Applicability

The standard is intended for use during the development and acquisition of "mission-critical" computer software, as defined in the related Department of Defense directive 5000.29. It specifies project processes, deliverables, control procedures, and a large variety of software development and quality assurance items, to be used by DOD contractors in the development and supply of software.

16.3 Features and Exceptional Points

a. This 90 page standard covers Scope, Referenced Documents, and Definitions in sections 1, 2, and 3 respectively. It then provides the bulk of its information in the next two sections. The 7 page section 4 General Requirements, describes and illustrates the software development life cycle. It also defines the specified software organization (see point c below), addresses in general terms the software quality assurance requirements (4.3), and addresses the issue of off-the-shelf software. The 32 page section 5 Detailed Requirements, provides the bulk of the specified SQA elements.

b. The standard states "Software development is usually an iterative process, in which an iteration of the software development cycle occurs one or more times during each of the system life cycle phases" (section 1.2). The standard has been designed to be applicable to each separate iteration.

c. The standard introduces the idea of Computer Software Configuration Items (CSCI's), Top Level Computer Software Components (TLCSC's), Lower-level Computer Software Components (LLCSC's), and Units.

d. The standard contains a number of appendices including further detail on software life cycles, and a 13 page appendix D on guidelines for tailoring the standard to different projects and development tasks.

e. Section 4.4 Use of Commercially available, reusable, and Government furnished software (mentioned above), and Appendix D both encourage the use of off-the-shelf products - a very important development. They do however also address the difficult problem of SQA requirements for such products. This is a good standard to study for those readers particularly concerned about this topic.

16.4 Strengths and Weaknesses

a. The standard is strong in listing software application areas where the standard is, and is not, applicable.

b. A strength of the standard is the fact that it addresses the use of commercially available, reusable, and Government-furnished software within the system being developed. It states the requirements for SQA on such software.

c. The document, as with most military SQA standards, makes extensive use of a terminology all its own. Whilst this allows for completeness and accuracy, it increases the magnitude of the task involved in understanding and working with the standard.

d. The abbreviations used throughout the document are mostly defined when first used. There is also a definitions section (3.) provided. However the many acronyms and abbreviations used are listed in appendix A. It would be better if this list was provided at the beginning of the document, as essential data, rather than in an appendix as "further detail".

e. The standard takes recognition of the currently dynamic nature of software engineering, and attempts to allow for selective application of the standard to specific projects. An appendix containing guidelines for tailoring the standard is supplied.

f. Along with most military QA standards, this document implements quality largely via the submission, by the

contractor, of documents (Data Items - DI's) to the contracting agency for approval. The standard specifies, in great detail, the production process, the contents and format, and relationships etc. of the various called-up data items. This method of ensuring quality should be of interest to the AECB, who are in a similar position with respect to their applicants.

16.5 Software Development Impact

As with most military SQA standards, this standard will have considerable influence on the software development process, both in cost and schedule, and also in management approach. It is likely that a major software producer would already have many of the required SQA items already in place, but conforming to this standard may well require him to revise the management structures, procedures and processes he is currently using in his software development and his quality assurance departments.

17. MIL-S-52779A, Aug 1979. Software Quality Assurance Program Requirements

17.1 Description of the Standard

This is a U. S. Department of Defence specification. It contains the requirements for a software quality assurance program to be established by all DOD software contractors, where such a program is called up in the contract specification documents.

17.2 Areas of Applicability

This specification applies to the acquisition of software (computer programs and related data and documentation) for all departments and agencies of the Department of Defense. The document specifically includes firmware, non-deliverable software, and software as a portion of a system developed under DOD contract.

17.3 Features and Exceptional Points

This military specification starts with a good description of the scope and applicability of the document. A good feature of this section is the explanation of the intent of the specification. The specification also states that it is intended that the SQA program be effectively tailored with other aspects of the contractors QA, administrative and technical operations.

The major part of the document is the section 3, Requirements, where the major subsections of the contractors SQA program plan document are called up. Whilst the descriptions are brief (one paragraph each), this specification dictates that the plan is to address such items as:

3.2.1 Tools, Techniques, and Methodologies.
3.2.2 Computer Program Design.
3.2.3 Work Certification.
3.2.4 Documentation.
3.2.5 Computer Program Library Controls.
3.2.6 Reviews and Audits.
3.2.7 Configuration Management.
3.2.8 Testing.
3.2.9 Corrective Action.

Section 4, Quality Assurance Provisions states that the contractor is required, notwithstanding this document, to meet the contract requirements, and reserves the Government's right to review the product at the contractor's site etc. if necessary.

Section 5, Preparation for Delivery calls up procedures for ensuring integrity of software during delivery etc.

Finally section 6, Notes, contains non-binding advice to Government contractual personnel, regarding the call up of SQA items in the bid and contract documents.

17.4 Strengths and Weaknesses

a. This five page document primarily specifies the items to be addressed in the contractor-provided Software Quality Assurance Plan. It makes little attempt to say how each of the called up items should be addressed by the contractor. Hence it is likely that Government contracting personnel would have to reference more detailed SQA standards and documents to obtain satisfactory software development.

b. The specified SQA Plan document called up does not require a contractor to:

   1) Issue or have in place any form of SQA policy statement committing the organization and personnel to the intent of the program.

   2) Identify, elements of his organization, his personnel, or their capabilities, within his SQA organization.

c. Section 3.2.5 Computer Program Library controls, and section 3.2.7 Configuration Management, would appear to overlap, as this author believes that library controls is a subsection of configuration management.

d. No mention is made of the need for testing subsequent to modification or change. This could be mentioned in section 3.2.8 Testing, 3.2.9 Corrective Action, or even in sections 3.2.5 or 3.2.7 above.

e. The specification does not require the contractor to report to the Government via status or progress reports. Even section section 3.2.9 Corrective Action, does not specify

the reporting of problems to other than "appropriate management levels" presumably within the contractors organization.

In conclusion, this document is a fairly brief, high level specification for the contents of a DOD contractor SQA Plan document.

17.5 Software Development Impact

It is not felt that this particular document would impact the software development process unduly. The impact does depend of course, on the contractor-provided detailed procedures to be used during the software development effort.

18. DOD-STD-1679A (Navy), Oct 1983. Software Development

18.1 Description of the Standard

This U.S. Navy produced military standard is intended to establish uniform minimum requirements for the development of software for the Department of the Navy.

18.2 Areas of Applicability

This standard is applicable to the development of software for the Navy and is available for use by all departments and agencies of the Department of Defense. It is felt that criticality of performance, changing operational requirements, and life cycle costs, require a standard specifically addressing Government software. Section 1.2 Application, states that these requirements apply when invoked in a specification or statement of work. It also states that the contractor must apply these requirements to his subcontractors.

It should be noted that this standard has recently been superceded by DOD-STD-2167 (reviewed earlier) but is still in use within the industry, and is also the current basis for much of the defence industry's SQA organization.

18.3 Features and Exceptional Points

The document starts with a good one-page foreword presenting the reasons for a specific Government standard. Those who argue that there is no requirement for the application of military-style SQA procedures in the commercial field, should be required to prove that the three factors listed are not applicable to their development!

Sections 1, 2, and 3 of the standard address the Scope, Reference documents, and definitions respectively. The seven paragraphs of the two page section 4 present the general requirements.

The bulk of the standard (29 pages) is section 5, Detailed Requirements. It contains the following sections:

5.1 Software performance requirements.
5.2 Software design.
5.3 Programming standards.

5.4 Programming conventions.
5.5 Software implementation.
5.6 Software generation.
5.7 Software operation.
5.8 Software testing.
5.9 Software quality assurance.
5.10 Software acceptance.
5.11 Software configuration management.
5.12 Software management planning.

As can be seen, this standard addresses the complete process of designing and developing software, and encompasses many management as well as quality assurance items.

18.4 Strengths and Weaknesses

a. The standard contains an implicit software development and maintenance life cycle, but should spell this out more clearly, i.e. describe the life cycle at the beginning and organize the document around it (c.f. DGMEM and DOD-STD-2167).

b. The standard is good in addressing the applicability to subcontractors involved in software development (e.g. 4.6 Subcontractor control), but does not address the subject of off-the-shelf development or runtime software.

c. Sections 5.3 Programming standards, and 5.4 Programming conventions are good in that they specify definite rules in these areas, but have the disadvantage that they are very language-type dependant (i.e. work well for CMS-2 but not so well for LISP). In particular section 5.4.11 Software listings, specifies machine code listings interspersed among higher level language statements in listings, a facility that is rarely available (but is with the U.S. Navy language CMS-2).

d. Section 5.4.13 Load maps, seems to have a line or portion of sentence missing "This mapping shall include ??? delineating all....".

e. The section 5.1 Software performance requirements, seems to specify good, complete functional analysis. It should in this authors view, be named somewhat differently (e.g. requirements analysis is in more general use - performance analysis is a more limited action). More seriously, it specifies items that could be very difficult to supply at

this early stage of a project, which tends to result in the mindless production of inaccurate estimates (i.e. 5.1.3 system resources).

f. Section 5.2 Software design, takes no recognition of the fact that it is current practice to consider design in two stages, i.e. high level design and detailed design. Also, although the section specifies the top down design and implementation process very well, it makes no allowance for complete or full detail on high-level modules, with lesser detail (possibly just functional description, etc.) for lower or bottom modules. This approach more correctly represents the accuracy of the real-world design process.

g. Section 5.10.2 Stress test requirements for software acceptance, contains very strong, detailed procedures and instructions for the stress test. This is good except that one can envisage a number of systems in which certain of these provisions would be difficult to accommodate exactly as worded e.g. saturation of data channels and internal buffers etc.

h. There are a number of areas in the standard where too much detail is provided, and where it might be better for the contractor to specify the detail level. For example, section 5.4 Programming conventions, and section 5.8.4 software trouble reporting. Also section 5.12 Software management planning, specifies items normally considered a contractor responsibility, e.g. 5.12.2 Computer resources requirements.

i. There are a number of places throughout the standard where specific numbers are mentioned that would be better supplied as part of the specific contract, i.e. 5.2.2.4 processing time reserves of 20%, 5.5.2 minimum 20% reserve capacity. This author feels the best approach is to mention the need for a reserve, then separately list the items to be supplied by the contract, thus forcing both the contracting authority and the supplier to address the issue, but allowing the actual number to be project dependent.

In conclusion, this standard is the most detailed and complete document examined in this study. It is more than an SQA standard however as it includes many management items. It is however in this authors opinion, too detailed and rigid to apply in the commercial field, and fails to allow for future change in software engineering techniques.

18.5 Software Development Impact

Of all the standards reviewed in this report, it is felt that this one would have the most impact, budget and schedule-wise on a software project. Due to the completeness, rigorousness, and inflexibility of the document, considerable extra activity is involved in the implementation of this software development standard.

D. EXTRACTION OF MAJOR REQUIRED ELEMENTS

This section presents the major elements of the various software quality assurance standards, practices and programs extracted from the previous section.

The section is arranged under the four subheadings Management, Engineering, Testing, and Operational and Support. The various extracted SQA items are listed under each heading, but note that in some cases the assigned subsection was somewhat arbitrary if a particular SQA item did not readily fall under one of the above four headings.

Each SQA item is briefly described. To provide full details is beyond the scope of this report. The reader is referred to the reference numbers following each SQA item for further details. These numbers refer to the section C entries in the Table of Contents.

Note that there are terminology and differences of approach etc. between the various reviewed standards, dependent on the purpose and audience of the standard. Hence it proved necessary to "translate" many of the elements from the various standards into a common terminology. This terminology is self-evident and so will not be further explained. However note that the terms "software developer" and "client" are used to refer to the two parties to a software purchasing, developing or approval agreement. From the point of view of the AECB, software developer refers to a group within a nuclear power utility or the utility's agents or subcontractors. Client would normally refer to the nuclear power utility, although via the regulatory process, the AECB would also play the role of a client from the viewpoint of reviewing information supporting licence applications.

1. Management

a. The software developer shall be responsible for planning, implementing and maintaining a software quality assurance program [1, 9, 16, 17].

b. The software developer may delegate to others part or all of the work of planning, establishing, and implementing the SQA program but shall retain responsibility for its effectiveness [1, 7, 17].

c. Senior management of the software developer, and his subcontractors in turn, shall issue a written policy statement which commits their respective organizations to implementing and maintaining an SQA program, and establishes its authority within the organization [7].

d. The software developer shall plan, implement and maintain procedures and controls that will promptly detect any non-conformance with the SQA plan [1, 7].

e. The software developer shall prepare and submit a Software QA manual and related procedures, for client acceptance prior to the start of software development [1, 9].

f. The software developer shall appoint an SQA Manager who has the necessary authority, and is independent of the development task, to be responsible for the SQA program [1].

g. The client shall undertake to review and accept the SQA manual and procedures of the software developer prior to the start of software development [1].

h. The software developer shall provide the client with access to his premises, and relevant information for review, surveillance or audit purposes [1].

•

i. The software developer shall make known to the client the responsibilities, authorities and organization of his SQA and development staff, plus indicate the relationship between positions [1, 9].

j. The subcontractors of the software developer are responsible to the software developer, and their sub-subcontractors similarly to the subcontractors, for establishing and implementing a similar SQA program [7].

k. The software developer shall prove appropriate independence between the various positions responsible for software development, SQA programs and procedures, and the various verification, review and audit functions associated with the SQA program [7].

l. The software developer shall use appropriately qualified

personnel in the performance of the SQA program procedures and items, shall undertake training etc. as appropriate, and shall keep relevant personnel records to prove the above [7].

m. The software developer shall conduct formal reviews of his SQA program periodically to determine its adequacy and effectiveness [7].

n. The software developer shall document the process flow by which functional requirements are eventually translated into delivered software and documentation [9, 10].

2. Engineering

a. The client shall ensure that the appropriate SQA standards and requirements are called up in the project tendering documentation [1, 7].

b. The software developer shall ensure the inclusion of the appropriate SQA standards and requirements in all subcontracts [7].

c. A comprehensive system of planned and documented audits shall be carried out to provide assurance that activities affecting quality comply with all aspects of the SQA program, and to determine the effectiveness of the program [7].

d. Documented procedures shall be established for control of non-conformance with the SQA program [1, 7].

e. Specific SQA procedures and documentation shall be in place to identify and record "configuration items". Any changes, faults etc. on configured items (documentation, software modules/programs, build and maintenance procedures, test data and procedures, software development environments etc.) shall be reviewed, documented, reported, and retained [7, 9].

f. The software developer shall be required to retain all records relating to the SQA program , its implementation, its effectiveness, and its operation. Such records shall be available for client audit and review purposes [7, 9].

g. The software developer shall produce a Requirements Specification document describing the detailed functional requirements of the software [10, 11, 14].

h. The Requirements Specification document should be subject to a review by both client and developer personnel, including SQA department staff, to ensure it completely and correctly describes the full requirements of the software [11].

i. The software developer shall produce a Design Description document detailing the software design,

including data bases, internal interfaces etc. [11].

j. The Design Description document shall be subjected to a review by both the client and software developer staff, including SQA department personnel to ensure the technical adequacy of the design, and to ensure that all requirements have been addressed by the design [9, 11].

k. The software developer shall be required to identify and specify all relevant standards, practices and conventions to be applied during software development. He shall also put in place procedures to ensure such standards are rigorously followed [9, 11].

l. The software developer shall be required to identify and document all interfaces, i.e. all the functional and physical characteristics relevant to two or more hardware or software modules, provided by one or more separate organizations [12].

m. The software developer shall be required to produce, via a configuration management plan, the detailed procedures and facilities necessary to track, control, store, and document, the various software, hardware and documentation items produced [5, 11, 12]

n. The software developer shall have procedures and controls to ensure corrective action is promptly taken to document and correct deficiencies [9].

o. The software developer shall identify and describe any special tools, techniques and methodologies used, for example, to document requirements, to produce code, to test requirements, to do special analysis, to aid in integration, commissioning or simulation etc. [9].

p. The software developer shall be required to have in place, and to document, procedures to ensure that subcontractor deliverables conform to requirements [9].

q. The software developer shall have, and describe, procedures for the production and control of "work orders" for developing software and other deliverables [9].

3. Testing

a. The software developer shall produce a Test Plan, detailing the scope, approaches, resources, levels of testing, and items to be tested. Such a document shall be produced in a timely manner such as to allow extensive review without adverse project impact [9, 11, 13].

b. The software developer shall produce a test specification describing test data, procedures, environment, resources and personnel etc. associated with every test to be performed on the software [13].

c. The software developer shall produce a test report detailing the results of all tests performed on the software [13].

d. The software developer shall document the procedures for building the system from the source code [10].

e. The software developer shall document the pass criteria for all tests prior to test execution [2].

f. The software developer shall prove the independence of the test appraiser from other vested interests within the developers organization [2].

g. The software developer shall prove the correctness of the test methods, tools etc. possibly by testing test code itself [2].

4. Operational and Support

a. The software developer shall produce a User Manual detailing complete information with regard to usage of the software, including operator commands, inherent restrictions and limitations, describing all error messages and fault conditions, and all external input/output actions [4, 10, 11, 15].

b. The software developer shall document and design the tests and procedures necessary to ensure that the in-service system continues to meet its requirements [2].

c. The system maintainer shall have in place procedures to monitor key aspects of the system (e.g. software, hardware, inputs, and usage) to ensure that the system requirements continue to be met [3].

d. The system maintainer shall have in place procedures to obtain, and retain periodic records such that continued, correct system operation can be monitored and periodically audited [3].

E. STANDARDS AND PRACTICES RECOMMENDATION

This section presents the recommendations of the report author with regard to software quality assurance practices considered appropriate and applicable to the nuclear power industry.

The recommendations are presented in two major parts. The first consists of general recommendations with regard to the application of software quality assurance standards within the industry. The second part consists of specific elements recommended for inclusion within any adopted or developed standard. For convenience of referencing this second section is organized under the subheadings of Management, Engineering, Testing, and Operations and Support.

1. General Recommendations

a. It is readily apparent that commercial software quality assurance standards have lagged considerably behind developments in other fields, in particular the military area. It is not so much that the actual processes and practices in use have been lacking, as that the detailed software engineering practices ultimately responsible for quality in produced software have not been mandated or standardized for the commercial area. A major reason for this has been the dynamic nature of software engineering in the last 10 years coupled with the time required to develop standards via the consensus method within the commercial sector. It is to be hoped that this situation will be improved in the near future.

b. It is recommended that the AECB pursue the development of software quality assurance standards as part of the CSA N286 series. In particular, there appears to be no reason that such developed standards could not be called up by the tier-one N286.0 Standard.

c. It is recommended that any standards developed include, possibly via appendices etc., checklists and suggested items for inclusion. These serve as information sources and tutorial material without being a mandated part of the standard. They also encourage developers to address all mentioned items.

d. It is strongly recommended that particular attention be paid to the subject of quality metrics for software, in the near future. Whilst this subject is not yet sufficiently mature for inclusion in SQA standards, and there remains many problems associated with their application to real projects, the potential is rapidly developing for specifying quantitative standards of quality for software systems [Refs 5, 6 and 7]. Certainly the earlier section B evaluation criteria items 2, 3 and 9, for example, would be easier proved if software quality metrics were available.

e. The ANSI/IEEE standards seem to be the best, most complete set of software engineering standards currently available in the commercial field. They are structured in a tiered fashion to allow partial callup and upgrading as necessary. However in a number of points they do not merge/agree correctly. For example, the Std 730 section 3.4.2.3 callup of an SVVP document does not agree with the document definitions of STD 829. It is therefore recommended that, as far as is possible, common terminology, approaches, life cycle phases etc. be used in linked upper/lower tier standards.

f. The use by applicants of vendor-supplied, subcontractor-developed, or off-the-shelf software, must be accounted for in future standards. It is recommended that for critical, safety-related systems:

   1. a general design principle of "minimum usage, and only of required features" be used.
   2. all included such software should be required to meet the same, or equivalent, SQA standards or reliability criteria as the developed software.
   3. the supplier/vendor should be required to have in place a full discrepancy reporting and update mechanism, such that this safety-related application can benefit from the experiences of other vendor-supplied users.

   In other words, since off-the-shelf products are preferred for cost saving, risk reduction, and proven performance (all of which is certainly true), it is suggested that suppliers be required to provide proof or evidence of reliability etc.

g. Similarly, it is recommended that the subject of verification and validation of support software be addressed. It is of little benefit if an applicant supplies beautifully designed, coded, tested and documented FORTRAN code, if a bug in a new version of

his compiler allows incorrect machine code to be included in a safety shutdown system.

h. Modern software systems are very much layered products, with final application code running, via a compiler or on-line interpreter, on top of operating system, data base, screen handler code etc., in turn running on top of machine language interpreting firmware and possibly peripheral, device or memory access firmware or programmable logic arrays etc. This layered approach means that each layer of the complete system must be guaranteed correct in some manner. This is a similar SQA problem to that provided by third-party or vendor-supplied software.

i. It is recommended that in all standardized documents, plans and specifications etc., if there is a required or suggested table of contents, that all specified sections be required. If a particular section is not applicable, it should be so stated and explained why, rather than merely being left out.

j. The IEEE standards contain a foreword which is technically not part of the standard. These forewords are a useful mechanism for providing background information and justification, and an explanation of related topics, to the standard. This author would recommend consideration be given to a similar mechanism on future standards, particularly for subjects not widely understood or accepted by all prospective standards users [c.f. foreword of IEEE 830-1984].

2. Specific SQA Element Recommendations

2.1 Management

a. Senior management of an applicant should be required to issue or have in place a specific policy statement with regard to the commitment of the organization to quality in the production of software. Such a policy statement should demonstrate to the AECB, and more importantly, to the applicants own staff, the company commitment to software quality assurance.

b. The applicant should be required to have in place a generic software quality assurance program applicable to all software produced by the applicant's staff.

c. The applicant should be required to submit a specific, project-dependent, software quality assurance plan drawn from the contents of the company SQA program. The plan should be approved prior to the start of software development.

d. The applicant should provide full details of responsibilities, authorities and organization of both his SQA department and his software development department, plus indicate the relationship between the various positions.

e. An applicant should be required to identify the individual responsible for development and maintenance of the SQA program (hereafter called "the SQA manager"), and indicate clearly his independence from the software development department.

f. It should be a requirement that all personnel within the SQA department be adequately and correctly qualified for their roles and responsibilities.

2.2 Engineering

a. It is recommended that all future SQA standards be developed with a specific software life cycle model in mind. Whilst there are many differences of nomenclature and detailed requirements among the standards analyzed earlier in section B, there is much general agreement on the software life cycle, and the life cycle is the basis for further standardization of documentation, reviews and audits, testing requirements, terminology etc.

b. It is very important that the various phases of the project life cycle mentioned above, be performed serially and independently. The output from one phase should serve as the input for the next or subsequent phases. Whilst in practice, some overlap is normal for staff utilization purposes, the principle should be that each phase is complete in itself.

c. It is very important that the functional requirements for a system be clearly and completely stated before design starts. If possible, use should be made of a structured requirements or design language in order to reduce to a minimum the ambiguities and mis-interpretations inherent in spoken languages.

d. It is recommended that the design and specification of tests be a separate responsibility from the code design and implementation. The level of testing (e.g. module, unit, integration, system etc.) at which this separation should apply is debatable. An extension of this philosophy is that each phase of the software life cycle should have some form of independent check, review or verification activity by some independent group; hence the use of preliminary and critical design reviews, the formal submission of requirements specifications etc.

e. It is recommended that an applicant be required to produce and implement, within his SQA program and plan, a configuration management system for software, hardware, documentation, and support environment etc. Where maintenance responsibility for the system is transferred from a developer (e.g. AECL) to a user or owner (e.g. Ontario Hydro), the maintainer should also have in place a configuration management scheme for code changes, updates, and enhancements.

2.3 Testing

Due to the specific mandate of the AECB with regards to safety, testing is a vital and important area in the proving of functional correctness in all systems submitted for licencing. This section contains specific recommendations for ensuring comprehensive testing of safety-related systems both during commissioning and during operation.

a. It is recommended that a subset of the total software life cycle, a testing cycle be developed. This testing sub-cycle should delineate the following:

   1. Prototype/Feasibility or Concept testing.
   2. Individual module unit testing
   3. Major module integration testing
   4. Complete system functional testing
   5. Commissioning testing
   6. In-service testing
   7. Audit testing.

   Generally speaking, each phase in the software life cycle should have a related testing aspect. An applicant should be required to specify the testing requirements prior to the phase being started.

b. An applicant should be required to perform individual module or unit testing as well as system testing. Experience has proven that for complex software-based systems, so-called "black-box" testing is inadequate for finding all software bugs. Testing each individual software program against the original program design specification is necessary [Ref 3, Future Methodology Improvements, Software Testability].

c. It should be a requirement that all tests be exactly repeatable, and have comprehensive, savable hardcopy output. An applicant should be required to prove the degree of completeness and accuracy of presented test results.

d. The applicant should be required to prove the independence of the test specification function from the software design and development function (i.e. The tests should be specified by a separate group).

e. All personnel engaged in specifying and running tests should be adequately qualified and trained for their

responsibility.

f. The question of verifying and validating the correctness of the functional requirements and the software design should be addressed by an applicant. Where the above items have been produced in a structured, machine readable form (there are a number of structured analysis languages available) it is possible to use machine-based evaluation and testing techniques [Ref 3, Future Methodology Improvements, Defining Specification Document].

g. It is recommended that any value calculations, mathematical models etc. implemented within critical software, be separately and independently designed and implemented in any related test bed situations. This requirement prevents errors in design or coding being duplicated in both testor and testee. Similarly it is wise to use, for example, different programming languages etc. This problem perhaps, is the software equivalent of common mode failure [Ref 4].

h. It is recommended that in any test standards prepared, some statement as to the minimum or desired range of test data for a software unit, module or system, be made. For example, representative values across the supported range, singular points, maximum and minimum parameter values, representative out-of-range values, maximum and minimum number of supported inputs etc.

i. It is recommended that testing should not be limited to merely testing correct, functional operation. That is, testing should not be limited to confirming correct implementation of requirements, but should also specifically confirm that no additional or unintended functions or limitations have been implemented that would adversely impact system operation.

2.4 Operational and Support

a. It is recommended that a design principle encouraging the use of built-in, automatic test features (e.g. memory checksumming, process and task monitoring, memory address checking etc.) be encouraged for safety-related systems. Such features should not be limited to attempting to detect hardware problems, but should also be designed to detect unusual or abnormal software operation.

b. Applicants should be required to formally report all software errors, problems, bugs and abnormal events. This reporting should be as soon as possible after occurrence, followed by periodic summaries. All such reports should be pursued to a satisfactory conclusion by the applicant.

c. It is the current practice in safety-related systems, which are not normally active very often, to perform tests, on a period related to the expected mean time between hardware failures. Whilst software does not degrade in a time-related fashion as hardware does, an applicant should be required to perform periodic verification or audit that the current running software configuration is as per last approved, i.e. that no untoward changes have been introduced to an in-service software system.

d. The question of the introduction of updates to an in-service system needs addressing. Ideally the full suite of functional tests, updated to reflect the functional change to be introduced, should be run. This is sometimes possible where such tests are performed via automatic facilities. But since such a requirement means a complete duplication of system commissioning, in some cases, it is not always feasible. However an applicant should be required to explain his plans/approach to updating in-service safety-related systems.

REFERENCES

1. NUCLEAR REGULATION - THE CANADIAN APPROACH Jon Jennekens, President AECB, AECB report INFO-0047 September 1981

2. ATOMIC ENERGY CONTROL BOARD: MODUS OPERANDI Blackman and Gummer, AECB report INFO-0083 February 1982

3. EXPERIENCE FROM QUALITY ASSURANCE IN NUCLEAR POWER PLANT PROTECTION SYSTEM SOFTWARE VALIDATION, Thomas & Reeves, IEEE Transactions on Nuclear Science, February 1980

4. SOFTWARE FOR FLIGHT CRITICAL DIGITAL ENGINE CONTROLS, Eccles, ASME Publication 1979

5. SPECIAL ISSUE ON SOFTWARE RELIABILITY, IEEE Transactions on Software Engineering, Dec 1985 & Jan 1986

6. Draft IEEE Standard P982, June 1985, Software Reliability Measurement

7. SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS Guidebook for Software Quality Measurement, RADC-TR-83-175 Vol II

BIBLIOGRAPHY

1. INTRODUCTION TO SOFTWARE QUALITY ASSURANCE 3rd Edition Richard Foster, private publication 1978

2. SOFTWARE QUALITY MANAGEMENT ed. John Cooper and Matthew Fisher Petrocelli Books 1979

3. QUALITY ASSURANCE FOR COMPUTER SOFTWARE, Robert Dunn and Richard Ullman, ISBN 0-07-018312-0, McGraw-Hill 1982

4. THE ART OF SOFTWARE TESTING, Glen Myers, John Wiley and Sons 1979

5. SOFTWARE RELIABILITY GUIDEBOOK, Robert Glass, Prentice-Hall 1979

6. SOFTWARE QUALITY ASSURANCE: A PRACTICAL APPROACH, Tsun S. Chow, A Tutorial from IEEE Computer Society Press, 1985, ISBN-0-8186-0569-3

APPENDIX A. ABBREVIATIONS

AECB	Atomic Energy Control Board
ANSI	American National Standards Institution
APEO	Association of Professional Engineers of Ontario
BS	British Standards (Institution)
CCB	Configuration Control Board
CSA	Canadian Standards Association
DGMEM	Director General Maritime Engineering and Maintenance
DND	Department of National Defence (Canada)
DOD	Department of Defence (U.S.)
FAA	Federal Aviation Authority (U.S.)
IEEE	Institute of Electrical and Electronic Engineers
NDHQ	Natinal Defence Headquarters (Ottawa, Canada)
QA	Quality Assurance
SCMP	Software Configuration Management Plan
SDM	System Development Methodology
SQA	Software Quality Assurance
SRS	Software Requirements Specification

APPENDIX B. ADDITIONAL STANDARDS AND RELATED DOCUMENTS

A number of additional standards, guidelines, and similar material was encountered in the course of undertaking this study, of which time did not permit a full examination. This material is listed below in order to provide additional information and further material for those readers interested in pursuing this topic further.

1. DLAH 8250.1, May 1981, Defense Logistics Agency, U.S. DOD. This is a handbook prepared for DLA staff involved in procurement quality assurance. It provides guidance where MIL-S-52779A is involved in contract SQA.

2. CAN3-N286.0-82, Quality Assurance Program Requirements for Nuclear Power Plants.

3. CAN3-N286.3-83, Construction Quality Assurance for Nuclear Power Plants.

4. CSA preliminary Standard N286.4-1979, Commissioning Quality Assurance for Nuclear Power Plants.

5. CSA preliminary Standard N286 5-1979, Operations Quality Assurance for Nuclear Power Plants.

6. AQAP-1 Edition 3, October 1983, NATO Requirements for an Industrial Quality Control System. This document supersedes DND 1015 as of April 1984.

7. TR-291, AECL, April 1985, Standards for tne Development of Scientific Research Software.

8. MIL-STD-483 (USAF) December 1970 Military Standard, Configuration Management Practices for Systems, Equipment, Munitions, and Computer Programs.

9. ASTM E623-83, Standard Guide for Developing Funtional Requirements for Computerized Laboratory Systems.

10. ASTM E624-83, Standard Guide for Developing Implementation Designs for Computerized Systems.

11. ASTM E1029-84, Standard Guide for Documentation of Clinical Laboratory Computer Systems.

12. 10CFR50 APPENDIX B, Quality Assurance Criteria for Nuclear Power Plants.

13. CSA Standard Z299.1 - 1985, Quality Assurance Program - Category 1.